Biography

James W. McIver, Jr. Age 46, married, three children.

Address: Chemistry Department, State University of New York at Buffalo, Buffalo, NY 14214.
mciver@sunybsc.bitnet phone: (716) 831-3629

Education: Johns Hopkins University BA (biophysics) 1963
University of Pennsylvania PhD (chemistry) 1966

Present Position: Associate Professor of Chemistry, SUNYAB

Research: My research interests lie in the areas of theoretical and computational chemistry. Current projects include the development of methodologies for computing potential energy surfaces for reacting systems and using these results to predict thermal rate constants in addition to molecular structural and spectral properties. Particular focus is on the use of parallel algorithms for implementing these problems on a concurrent computer.

Recent Publications

1. On the SCF Calculation of Excited States. M. McCourt and J. McIver, J. Comp. Chem. 8, 454 (1987).

2. Symbolic Algebra Software. A Useful Tool for Code Developers. M. McCourt and J. McIver. QCPE Bulletin 7, 69 (1987).

3. Spin Orbit Coupling in Biradicals. Ab Initio MCSCF Calculations on Trimethylene and the Methyl-Methyl Radical Pair. L. Carlacci, C. Doubleday , T. Furlani, H. King and J. McIver, J. Amer. Chem Soc. 109, 5323 (1987).

4. New Formulas for Integrals in Semiempirical Molecular Orbital Methods I. General Theory and Results for Two Center Overlap Integrals and their First and Second Derivatives with Respect to Cartesian Displacements. M. McCourt and J. McIver, J. Mol. Structure (Theochem), 163,343 (1988).

5. New Formulas for Integrals in Semiempirical Molecular Orbital Methods II. Coulomb Integrals and their First and Second Cartesian Derivatives in the NDDO Theory. J. Denton, M. McCourt and J. McIver. J. Mol. Structure (Theochem), 163,355 (1988).

6. Dynamics of Singlet Biradicals. Application of Ab-initio Variational Transition State Theory to the Fragmentation of Tetramethylene. C. Doubleday, M. Page and J. McIver. J. Mol. Structure (Theochem), 163, 331 (1988).

7. On Evaluating the Reaction Path Hamiltonian, M. Page and J. McIver. J. Chem. Phys. 88, 922 (1988)

POSITION STATEMENT 1.

Problems:

1. A lack of a clear definition, understandable to the ordinary scientist or engineer, of what exactly symbolic computation is. What is its general scope? What is the current state of development? What are the current limitations?

2. A lack of a resource whereby a scientist or engineer can find out whether or not his problems are suitable for symbolic computation, and if so, how to go about solving them using existing software.

Possible solutions:

1. Establishment of a national centralized facility. A good idea in that it would provide a resource and (hopefully) some standards. A bad idea in that it would be expensive, thus taking funds away from other possibly more effective means of promoting applications.

2. Promote the establishment of "applications" workshops, such as the short courses found at national meetings of the American Chemical Society.

3. Promote the development of local expertise (eg at research universities) and hardware and software facilities for performing symbolic computations. NSF might encourage universities to apply for funds to partially support such facilities, for example.

Position Statement 2.

I have only used MACSYMA.

Problems: There are lots of little problems with MACSYMA (can't do gradef on array elements, etc.) but a really serious one is that the software and its documentation reflect its origins. It is almost incomprehensible to users who don't know lisp. Some of the examples are helpful, but for the most part they are not clearly explained. One frequently resorts to trial and error which is time wasting and often frustrating. Another problem is its relative machine dependence. It only runs on VAX's (very poorly when lots of users are logged on) and a few workstations (it runs on a Sun 3 but not on a Sun 4).

Possible solutions:

1. Let the people who are commercializing this software (who must certainly be aware of the problems) worry about it. The problem here is that their goals (which have to do with quarterly earnings statements) do not always coincide with those of users or potential users.

2. Get some competition. Perhaps other systems don't have these problems. Can NSF fund developers who intend to commercialize their systems?

3. Encourage the growth of vendor independent user groups. This at least might result in better documentation of the software and its limitations in addition to their more traditional roles.

POSITION STATEMENT 3.

I KNOW that my formula can be written as a rational expression involving a short sum of factored terms. Why wont the software make it look like I want it to?

Position statement 4.

Problems: (again, I only have experience with MACSYMA)

1. MACSYMA is weak in the area of producing code for numerical computation. It can only produce FORTRAN assignment statements. It cannot, for example produce loops with an optimized structure. Horner's rule seems to be the only optimization tool available.

2. MACSYMA cannot do linear algebra, a serious shortcomming for much numerical work (see the article by Gentleman).

Solution: Encourage research in this area. Interdisciplinary work will be key here. Numerical analysts, scientists and engineers should work with the symbolic computation developer.

Position statement 5.

As implied by the notes by Sweedler, dissemination is the real problem with symbolic computation.

1. No data. Most of the discussion of symbolic computation (pro or con) is testimonial. How many users and developers are there? How are they distributed across disciplines? Where is it being taught, and by who? Has its use been evaluated? (ie what are the results of teaching calculus at Waterloo by Maple compared to traditional methods?) I have never seen a survey or even a question on a survey asking about symbolic computation, although I get lots about computers, compilers and other types of software.

2. "Bring your problems to me". Clinics are a nice way for companies to get their problems solved by somebody else, but do they really help get symbolic computation to the workplace? It seems better to have employees use it while they are students. They will see that it gets used after they graduate. Moreover, dont worry about it not being taught in math and computer science courses. It is better to have it used in science and engineering courses where it can be used on real problems. Symbolic computation people should get out into other departments to promote its use in courses. Dont wait for them to come to you.

Position Statement 6

Topic: Underfunded research in Symbolic Computation

Problem:

Symbolic Computation seems to be an orphan. Researchers are few in number and have trouble getting their proposals funded. Mathematicians and computer scientists are perceived as somewhat hostile peer reviewers.

Solution:

Capitalize on the great strength of Symbolic Computation, its broad interdisciplinary appeal. Weasel your way into one of the grossly overfunded areas such as supercomputing, materials science, AIDS research, etc. Every campus has at least one of these. Find somebody from another department who might profit from your research and write joint grant proposals. Funding agencies should very strongly encourage such interdisciplinary activities involving Symbolic computation research.

Symbolic Computation

A. Nerode

I view symbolic computation as a natural historic outgrowth of mathematical and scientific practice. Symbolic computation enriches each branch of mathematics by discovering and refining its computational algorithms.

First, symbolic computation is a service to science and engineering. To the extent that its algorithms are successful, it serves to make each branch of mathematics potentially applicable to areas of science and engineering nature where that branch of mathematics supplies an implicit functional description to which an explicit symbolic solution may be useful for insight and prdiction.

Second, symbolic computation is a service to pure mathematical science. It provides means for carrying out both counterexamples to previously attractive conjectures and also proofs (such as the four color theorem) which are too long for the patience and even the lifetimes of individual mathematicians.

Third, development of algorithms goes hand in hand with the development of concepts to deal with them. Witness Gauss's theory of binary quadratic forms, entirely computationally based, giving rise to ideal theory, or Galois's purely computational introduction of Galois groups for computing solvability by radicals giving rise to group theory. The concepts that such procedures lead to may not be discoverable any other way in pure mathematics. The algorithms based on the datatypes these concepts introduce may not have any other avenue for discovery. Symbolic computational algorithms are a fertile source of mathematical ideas.

I view the development of symbolic algorithms for each branch of mathematics as an area with its own gradually emerging coherence. In what common mathematical terms can this coherence be expressed? That is, is there a common underlying mathematical theory of symbolic algorithms?

One point of view is that symbolic computation is mathematically a theory of rewrite systems. Expressions are reduced to simpler form by reduction steps which lower a rank, the rank being an ordinal. Sometimes the object is to normalize by defining termination of a reduction process, that is what it means to stop with an acceptable standard form. Such is the Buchberger theory of standard bases for ideals. Sometimes one rewrites when there is no definition of a finite standard form and termination, such as when developing as many terms of a power series as you happen to want by a procedure which would produce all terms in infinite time.

It is possible to view declarative programming and automatic theorem proving for classical logic in this light. For example, the resolution method of automatic theorem proving (exemplified by SLD resolution for PROLOG) begins by reducing to clausal form (a terminating procedure, a standard form), then searches for a derivation of a goal from hypotheses. If this search succeeds, there is termination and a derivatiomn If not, the search for a derivation can go on forever.

It is possible to view applicative programming in this light. There are two classes of programming languages for applicative programming. Somebased on the typed lambda calculus, and some are based on the untyped lambda calculus. In the typed theories all computations terminate, in the untyped unending computations are possible. A more refined view than that of rewrite systems is to regard symbolic computation as the study of specific applied lambda calculi.

There is a deep underlying theme connecting declarative and applicative languages, the Curry–Howard isomorphism between constructive logical systems and lambda calculi. This expresses the idea that deduction and computation are really identical.

I think that the underlying mathematics of algorithms in each special area may well in the end be best expressed as normalization and strong normalization theorems for specific applied lambda calculi and combinatory algebras, and their corresponding logics. This leads naturally to implementation. Both typed and untyped lambda calculi have been implemented and may be useful as universal symbolic computation languages. Such languages as ML and MIRANDA and Constable's NUPRL are candidates for successors of LISP.

On the implementation side, I would like a common programming language with a clear semantics and all the useful constructs which covers both the declarative and the applicative side in a uniform way, allowing any sort of generic modular combinations of applicative and declarative programming as is characteristic of polymorphic language programming.

As is evident above, I take a broad view of what should comprise the subject of symbolic computation, namely that it should contain both the algorithms of the specific mathematical subjects and those of the related lambda calculi and logical systems..

Brief Biographical information:
Richard J. Fateman
Ph.D. in Applied Math, Harvard, 1971,
taught at MIT (math) 1971-74,
1974- present at Univ. Calif. Berkeley (Computer Science)
currently Computer Science chairman.

heavily involved in the early development of the Macsyma system at MIT. Former chair of ACM-SIGSAM.

Position Statement 1 (major problems with applying technology) Consider the variety of Computing systems:

Macsyma (Symbolics Macsyma, DOE/NESC Macsyma,Vaxima)
Reduce
Scratchpad II
Maple
SMP
Mu-math
Mathematica
Milo, Mathscribe, Expressionist
TeX, etc

Here are some problems. (pardon my shooting from the hip... I'm leaving out many qualifiers that might be put in here).

* Lack of rock-solid, efficient, well-designed and documented easy to understand and use systems.
* Premature commercialization of research systems resulting in proprietary source-code and parallel development of new algorithms.
* Poor environments for integration of software (exception: LISP machines - but they have other problems like inadequate Fortran, high cost).
* Some problems caused by attempts to run on low cost hardware.
* Lack of Education

Solutions: fund projects with appropriately structured mathematical and software engineering designs for algebraic system building. Promote standards -e.g. Common LISP, UNIX, X-windows, IEEE floating point. Insist on public disclosure of algorithms and programs in a re-usable form. Define new standards - e.g. for expression interchange. Identify interfaces of SAC with applications in engineering - e.g. optimization, finite-element code generation, "knowledge-engineering", robotics. Insist that new designs address a set of benchmark requirements.

Position Statement 2 (Major problems with systems)

See position statement 1.

Poor engineering. Either overly-restrictive programming (SAC-2, maybe Scratchpad II) or lack of attention to details of domain errors (division by zero, geometric inconsistencies) - (Macsyma, Reduce, SMP , probably everything else, too....)

Poor interfaces to numerical, graphical programs. (This is being worked on in various places).

Position Statement 3 (theory and algorithms problems)

(I expect this to be very controversial)

Probably NOT Grobner basis calculations (too slow for solving any problems of real interest)

Approximation (Practical Taylor serie, Pniseaux series, Newton-polygon procedure, expansion in other orthogonal functions, continued fractions)

Simplification and manipulation of special functions.

Probably NOT the incremental building of Risch-type integration for the next n differential equation or integral extensions.

Improved data structures and algorithms for parallel symbolic math - probably look for large problems in celestial mechanics/special purpose systems.

Probably NOT new hardware (e.g. systolic algebra).

Let NSA pay for cryptography, factoring, coding theory.

NSF might some of the inherently inapplicable theory into Math division.

Position Statement 4 (combining S&N)

There is no major problem except getting smart people who know a great deal about both areas to put the pieces together. Projects exist at UC Berkeley, University of Illinois, Kent State, University of Waterloo, NASA Ames, and probably other places. Not much money, or money spent where the smart people are not.

Such collaboration could significantly change the nature of both Symbolic and numeric processing (e.g. Genleman's paper).

How to improve the situation:

Buy a few sharp people and put them together for a year.

Position Statement 5 (awareness and use)

Introduce SAC into the general curiculum in Math and Engineering in Colleges and high schools.

targets: Calculus or pre-calculus algebra - maybe physics, maybe intro CS- (graphing, differention).
Problem is the language for math .ne. language for SAC.

Do a really bang-up job on SOMETHING visible.

E.g. in replacing a major integral table with a wholly superior program. Hardware technology is clearly available. (Must use lookup - Risch is just not good enough and never will be; probably would have to know more geometry than is common.)

BIOGRAPHY David L. Elliott

David L. Elliott is Program Director for System Theory in the ECSE Division, National Science Foundation; he is on leave from Washington University, St. Louis, where he is Professor of Mathematical Systems, specializing in nonlinear and geometric methods in Control System Theory. His Ph.D. is in System Science, UCLA, 1969; prior to that he was a mathematician in a Navy laboratory (now USNOSC, San Diego). His NSF program supports research in the design and analysis of control systems as well as related areas involving the application of mathematics to dynamic problems in engineering. Phone: (202)-357-9618; electronic mail: (Internet) delliott@NOTE.NSF.GOV (Bitnet) delliott@NSF.

Applications in Control System Analysis and Design
David L. Elliott, NSF

1. Transfer-function matrix analysis

In the design of multi-input multi-output linear control systems, the "transfer functions" are (rectangular) matrix functions of the complex variable s; usually the matrix components are rational functions of s, so the desired manipulations are just matrix algebra over the field of rational functions of one variable. In some cases the underlying mathematical structures involve the ring of polynomials (or formal power series) in s, and modules over these rings rather than vector spaces over a field. More recent work involves rational functions in two or more commuting variables ("2-d system theory" is used in picture processing), or replaces the field of complex numbers by a finite field (algebraic coding theory). Typical tasks are the reduction of matrices to echelon form, matrix factorization with various restrictions on the factors (i.e. Hurwitz stability), the solution of the Bezout equation, and routine computations of transfer matrices from block diagrams via matrix products and inverses. I would be interested in hearing about software which reliably implements these operations and could be made available to system theorists.
This may be an area in which several NSF programs can get together to further exploratory research.

2. Vector-field Lie algebras and differential forms in nonlinear system theory

The use of Lie algebra in control theory is somewhat analogous to its use in physics (for example the formulae of Campbell-Baker-Hausdorff, Magnus, Wei & Norman, etc.). REDUCE scripts involving jacobi brackets of vector fields--[f,g] = fg - gf -- have been written by Marino (Rome) for robot-arm theory, and Blankenship et al wrote similar MACSYMA packages for nonlinear filter design (System Research Center, University of Maryland.) Typical tasks are the computation of iterated jacobi brackets, determining dimension of linear spans of vector-field families, reduction to echelon form (Frobenius' Theorem in differential geometry), and the solution of systems of first-order linear partial differential equations. The coefficient functions in some applications (robotics, attitude control of satellites, electric power systems) involve only polynomials and trigonometric functions of x, y, ..., so the computations are finitistic; an appropriate algebraic context is differential algebra (in the manner of Ritt and Kholchin), which itself is being applied by Michel Fliess and others in France to such problems as finding the left or right inverse of a nonlinear system. The most well-known application of differential-geometric methods in control theory is the reduction of certain nonlinear systems (e.g. robot arms) to linear form by the use of nonlinear coordinates and nonlinear feedback; the computations involve vector fields and differential forms. Differential forms are objects dual to vector fields; symbolic computation with them was pioneered in the CARTAN package written for Apple computers by Hugo Wahlquist of CalTech JPL; more recently R. B. Gardner and colleagues (Chapel Hill) have begun developing symbolic-algebra software for a related approach to the classification of nonlinear systems. I would like to find out about SAC systems which would be particularly appropriate for this work, and about the experience of others in this area. --
---

A Workshop on Symbolic and Algebraic Computation:
Directions for Future Research

Name Shreeram S. Abhyankar Date 20 April 1988

BIOGRAPHY

Shreeram S. Abhyankar was born on 22 July 1930 in Ujjain in Central India. After his schooling and two year college studies in Gwalior and the last two years of college studies in Bombay, he obtained his bachelors degree from Bombay University in 1951. From there he proceeded to Harvard University where he obtained his doctoral degree in 1955 under the guidance of Oscar Zariski. In his thesis, Abhyankar solved the problem of resolution of singularities of algebraic surfaces for nonzero characteristic. Ten years later, he solved the same problem for arithmetical surfaces and for three-dimensional varieties for nonzero characteristic. After teaching in Columbia, Cornell, Princeton, Harvard, Johns Hopkins and Yale, Abhyankar moved to Purdue University in 1964. Since 1967, Abhyankar has has been Distinguished Marshall Professor of Mathematics at Purdue, where he was also appointed as a Professor of Industrial Engineering starting in September 1987, and as a Professor of Computer Science starting in April 1988.

In 1978, Abhyankar was awarded the Chauvenet Prize of the Mathematical Association of America for his article "Historical Ramblings in Algebraic Geometry and Related Algebra" that appeared in the June 1976 issue of the American Mathematical Monthly. In that article, Abhyankar said that his father initiated him to mathematics and Sanskrit poetry by teaching him portions of Bhaskaracharya's treatise on algebra called Beejganita, which was writtten around 1150 A.D. In 1976, Abhyankar was also instrumental in founding the Bhaskaracharya Pratishthana, which is a research institute in Pune, India, named after this great mathematician who was the director of an observatory in Ujjain.

Abhyankar is the author or coauthor of more than 80 articles, and he has published eight books including Ramification Theoretic Methods in Algebraic Geometry, Resolution of Singularities of Embedded Algebraic Surfaces, Algebraic Space Curves, and Enumerative Combinatorics of Young Tableaux. He has given numerous lectures at universities and conferences in Australia, Canada, East Germany, France, Greece, Holland, Hong Kong, India, Italy, Japan, Singapore, Spain, Taiwan, U.K., U.S.A., U.S.S.R., and West Germany, on his research in algebraic geometry, commutative algebra, local algebra, theory of functions of several complex variables, quantum elctro-dynamics, circuit theory, combinatorics, comuter-aided design, robotics, and geometric modeling. He is a member of the American Mathematical Society, Mathematical Association of America, Indian Mathematical Society, Phi Beta Kappa, and a Fellow of the Maharashtra Academy of Sciences, Indian National Science Academy, and Indian Academy of Sciences.

In recent years, Abhyankar has been taking an active interest in promoting the interdisciplinary impact of algebraic geometry on various application areas in Science and Engineering.

A Workshop on Symbolic and Algebraic Computation:
Directions for Future Research

Position Statement 1

Name (optional) Shreeram S. Abhyankar Date 20 April 1988

Topic: Major problems with applying the technology of symbolic and algebraic computation. What are some possible solutions? Where appropriate, indicate what actions NSF or others might take to improve the situation. Be as specific as possible.

Until about thirty years ago, people used to learn the two subjects of Analytic Geometry and Theory of Equations in high-school and college. Then these subjects vanished from the syllabus. Analytic Geometry first became a chapter and then a paragraph and then only a footnote in books on calculus. This was coupled with the higher and higher abstractions of Modern Algebraic Geometry to the point of inaccessibility. The resulting downgrading of algorithmic techniques, the very techniques which can be used in symbolic and algebraic computations and the dominance of existential proofs, which cannot easily be used for computations acts as a stumbling block for progress in computation. A major effort to revive and increase the awareness of a need for an algorithmic approach should help in developing computational tecniques. NSF clearly can support such efforts.

A Workshop on Symbolic and Algebraic Computation:
Directions for Future Research

Position Statement 2

Name (optional) Shreeram S. Abhyankar Date 20 April 1988

Topic: Major problems with software systems for SAC. What are possible solutions? Where appropriate, indicate what actions NSF or others might take to improve the situation.

Interdepartmental jealousy between departments of mathematics and computer science is preventing the two from working with each other to the advantage of both. An awareness of the power of various computer systems is arriving to the mathematical community in a very unfocussed way. Similarly, many mathematical results which could be of use in the design of software systems are not generally known to the system designers. NSF can encourage interaction.

A Workshop on Symbolic and Algebraic Computation:
Directions for Future Research

Position Statement 3

Name (optional) Shreeram S. Abhyankar Date 20 April 1988

Topic: Major problems in the theory and algorithm area of SAC. What are possible ways to attack the problems? Where appropriate, indicate what actions NSF or others might take to improve the situation.

Algebraic Geometry, at least in its classical form, is an amalgamation of Analytic Geometry and Theory of Equations. In this concrete viewpoint, algebraic geometry is the study of finite systems of multivariate polynomial (or power series) equations. The basic tools are transformations which are viewed as substituting new variables for the old and observing the effect this has on the combinatorial structure of the original polynomials and on the solutions of the given equations. Much of classical algebraic geometry is algorithmic in nature. It calls for constructive solutions. This approach is eminently well suited for progress in several significant mathematics problem areas such as Resolution of Singularities, Geometry of Affine Spaces, Elimination Theory, Invariant Theory, and so on. Theoretical algorithms are a precursor to computational algorithms, and hence the affinity of Algorithmic Mathematics with Computer Science and Engineering.

Now the neglected art of Algorithmic Algebraic Geometry is being resurrected by the scientists and engineers because of its increasingly important impact on various application areas, such as Computer Aided Design (CAD) and Computer Aided Manufacturing (CAM), Geometric Modeling, Symbolic Algebraic Computations, Modular Computations, Combinatorics, Robotics, Finite Element Method, Control Theory, and Optimization. This has been accelerated since the advent of high speed computers that enable us to carry out more and more complicated symbolic and numerical computations. Moreover, these application areas are in turn posing fundamental open mathematical problems in algebraic geometry, thus making it a two way street.

Question Why am I talking about algorithmic algebraic geometry in a workshop about symbolic algebraic computations?

Answer Because algorithmic algebraic geometry has always been my main field of interest and beause I believe that it is destined to be one of the principal customers of symbolic algebraic computations. A major link for this relationship is provided by the theory of Groebner Bases. Groebner Bases deal with constructive ideal theory, which is a movement started by Grete Hermann around 1926. Constructive ideal theory is urgently needed for progress in the problem of resolution of singularities in non-zero characteristic, which is a major area of concern in algebraic geometry. Another link between symbolic algebraic computations and algebraic geometry is provided by elimination theory and resultants. It is hoped that the techniques of symbolic algebra can be used to develop the machinery of multivariate resultants. This should be of great benefit in dealing with intersections of curves and surfaces and for attacking several problems in algebraic geometry such as the jacobian conjecture.

A Workshop on Symbolic and Algebraic Computation:
Directions for Future Research

Position Statement 4

Name (optional) Shreeram S. Abhyankar Date 20 April 1988

Topic: Major problems in combining symbolic and numeric computation. Where appropriate, indicate what actions NSF or others might take to improve the situation.

The dominance of calculus (most of it of a cook-book variety) has resulted in the elimination of much mathematics which is intellectually appealing and is also of great application value to science and industry. The subjects of analytic geometry, theory of equations, projective geometry, etc. have been removed from the college curriculum. There is a need to prepare readable material to make these subjects accessible to those to whom they were not taught, but now find a need to understand them, as well as to reintroduce an algorithmic algebraic orientation into the teaching of college mathematics. Supplementing the techniques of calculus with a rejuvanated algorithmic algebraic approach should go a long way in combining symbolic and numeric computation.

A Workshop on Symbolic and Algebraic Computation:
Directions for Future Research

Position Statement 5

Name (optional) Shreeram S. Abhyankar Date 20 April 1988

Topic: Major problems in developing awareness and use of SAC. What are possible ways to attack the problems? Where appropriate, indicate what actions NSF or others might take to improve the situation.

The artificial barriers between the various departments of science and engineering, and the resulting lack of facilities and opportunities for researchers from these compartments to meet each other, has delayed the awareness by mathematicians of the increasing use of Algorithmic Algebraic Geometry and Symbolic Algebraic Computations by a growing number of engineers. This has retarded what clearly promises to be a very beneficial partnership between mathematicians and engineers. NSF could support greater cooperation for an interdisciplinary environment. The evolving interdisciplinary nature of our subject is examplified by the fact that, during the last ten years, symbolic algebra and algebraic geometry papers are no longer just in mathematics journals but also in several engineering journals as well. More needs to be done to make students and faculty aware of the progress that has been made and how to take advantage of it in their own work. Many mathematicians are still computer novices, and a greater publicity effort is needed to show them the power of the new computer techniques. It must be made easier for students and faculty in computer science and engineering to learn about advances in mathematics, since these often could have a direct impact on their work. Seminars and institutes which include participants from a broad range of scientific backgrounds could spread the awareness of Algorithmic Geometry and Symbolic Algebraic Computations.

A Workshop on Symbolic and Algebraic Computation:
Directions of Future Research

Position Statement 6

Name (optional) Shreeram S. Abhyankar Date 20 April 1988

GENERIC TOPIC SHEET

Topic:

Sidelights. Discussion. Recommendations. What are possible ways to attack the problem? Where appropriate indicate what actions NSF or others might take to improve the situation.

I guess I am simply a roving ambassador who has a keen desire for mathematics in general, and algebraic geometry in particular, not to stay in its ivory tower because I am convinced that interdisciplinary contact between mathematics and various application areas in science and engineering is very relevant and mutually quite beneficial. This is certainly the case vis-a-vis algorithmic algebraic geometry.

The growth in size of universities and mushrooming specializations makes it difficult for people of common interests who are in different departments and schools to interact. Facilities for such interactions are urgently needed. There is a need to spread knowledge Algorithmic Algebraic Geometry and Symbolic Algebraic Computatiuons to the mathematics community, faculty as well as students. An awareness that by concrete calculations (symbolic as well as numerical) and by the use of graphics we get new insights and foresights into mathematical truths needs to be systematically spread.

NSF should support multidisciplinary teams which should be large enough to create a critical mass capable of making a major impact on Algorithmic Algebraic Geometry and Symbolic Algebra. They should include graduate and undergraduate students, as well as visiting scientists from various educational institutions and industrial research establishments. They could develop computational tools and methodology and to train future researchers so as to facilitate the multifaceted applications of Algorithmic Algebraic Geometry and Symbolic Algebra to science and engineering.

Short Biographical Sketch
Philip Wesley Smith

Date of birth: November 28, 1946
Place of birth: Gainesville, Florida
Marital status: Married, 2 children

Current business address

IMSL, Inc.
2500 ParkWest Tower One
2500 CityWest Boulevard
Houston, Texas 77042-3020

In the profession

- July 1985–present Manager, Mathematical Software Design Dept., IMSL
- September 1980–June 1985 Professor, Old Dominion University, Norfolk, VA 23508
- July–August 1981 SFB Visitor, University of Bonn
- January 1980–June 1980 Visiting Scientist, IBM T. J. Watson Research Center, Yorktown Heights, New York 10598
- September 1979–December 1979 Visiting Research Professor, University of Alberta, Edmonton, Alberta, Canada
- September 1976–September 1979 Associate Professor, Texas A&M University
- October 1972–September 1976 Assistant Professor, Texas A&M University

Degrees

- Ph.D. in Mathematics at Purdue (1972) Michael Golomb, advisor, thesis topic $W^{r,p}$-Splines.
- M.S. in Mathematics at Purdue (1970).
- B.A. in Mathematics at the Univ. of Virginia (1968).

Position Statement 1
Applying the technology

- Learning another programming language is a hinderance to applying SAC technology.
- Symbolic computations generate complicated results from simple-looking problems. Perhaps more intelligence in the design could reduce this problem.
- One must become an expert to utilize the system effectively on "real" problems.

Position Statement 4
Problems combining symbolic and numeric computation

- Identification of problems where interaction makes sense.
- If both numeric and symbolic systems are to interact and run simultaneously, then it is necessary to worry about effects of the operating system, compiler, etc.
- Symbolic systems do not manipulate the objects which most numerical analysts and engineers work with every day. These objects include splines, block matrices, canonical matrix factorizations, etc.

Position Statement 5
develop awareness and use of SAC

- It is necessary to demonstrate the clear superiority of SAC in at least one important engineering area. If this could be done then there would be no doubt as to the viability of this technology. Here is where the NSF could bring to bear some of its resources. Grants for feasibility studies in merging engineering applications and symbolic computing should be pursued.
- Access to good computing equipment (modern workstations) is essential.
- SAC systems are not generally available to either professional mathematicians or engineers. One method of generating interest in this technology is to develop math courses based on SAC. Obviously, the NSF has a role here.

Position Statement 6
General topics

- The successes of SAC must become more generic for SAC to move into the forefront of computing. Real engineering and computing problems must be demonstrated to need SAC for their solution. That is, it is not sufficient to show that SAC is marginally better than FORTRAN for a given problem. It must be an order of magnitude better.

- The NSF and educational institutions must make available the software and hardware to students in a "nonthreatening" environment.
- More graphical input and output for SAC is needed, as is provided in a modern workstation interface with mice and menus.

Moss Sweedler April 17, 1988

Topic: Major problems in the theory and algorithm area of SAC.

Like the Hydra, SAC research[1] has many faces. SAC research:

Utilizes results and techniques of **pure mathematics.**

Benefits from computer tests and trials, like an **experimental science.**

Is motivated and utilized, like an **applied science.**

Although the mathematical aspects of SAC research are **applied mathematics**, they are so different from traditional applied mathematics that applied math departments are not a natural home for SAC research. Nor are traditional applied mathematicians qualified reviewers for SAC research proposals.

Few institutions can provide more than a limited environment for SAC research. In academia, SAC is often the stepchild of computer science and mathematics departments, not receiving adequate priority or funding from either department.[2] It is common for **INTER DISCIPLINARY** areas not to receive adequate priority or funding from the **DISCIPLINES** they **INTER.** SAC research in academic mathematics departments may miss applied and experimental elements and SAC research in computer science departments may miss pure mathematical elements. The best (only?) solution is to establish SAC research as a separate area. This applies to funding agencies as well as academic institutions and may be necessary for SAC research to receive adequate funding. More funding is needed!

A national SAC center would be able to support all aspects of SAC research.

Computer algebra clinic programs[3] run by mathematics departments, would foster more of an applied orientation within mathematics departments, as well as promote undergraduate awareness of SAC. Let NSF solicit grants to launch or enhance computer algebra clinics.

---

[1] On algorithms for computation and the theory related to these algorithms.

[2] This may also be the case at NSF and other funding agencies.

[3] Described in (K) and (AA) in the: Notes on computational and computer algebra for U.S. science and engineering, in the preliminary reading material.

Moss Sweedler April 17, 1988

Topic: Major problems in developing awareness and use of SAC.

Were general purpose symbolic computation systems (GPSCS) to be commonly and casually used in undergraduate mathematics, science and engineering courses, widespread professional use of GPSCS would ensue. The current occurrence of GPSCS in undergraduate education is quite the opposite. Through undergraduate education programs NSF could encourage the ubiquitous use of GPSCS.

Computer algebra clinic programs[1] would promote undergraduate awareness of SAC. Let NSF solicit grants to launch or enhance computer algebra clinics.

A national SAC center would, by its very existence, draw national attention to SAC. This is not enough. The national SAC center must have or coordinate: workshops, course development, course dispersal, clinics, applications, research, etc. This will be the most effective way to spread SAC awareness.

---

[1] Described in (K) and (AA) in the: Notes on computational and computer algebra for U.S. science and engineering, in the preliminary reading material.

Moss Sweedler April 17, 1988

Topic: Major problems with applying the technology of symbolic and algebraic computation.

General purpose symbolic computation systems (GPSCS) utilize SAC technology. With the eventual widespread scientific and engineering use of GPSCS, this application of SAC technology will have immense impact. Were GPSCS to be commonly and casually used in undergraduate mathematics, science and engineering courses, widespread professional use of GPSCS would soon occur. For more along these lines, see my sheet: Major problems in developing awareness and use of SAC.

There are likely a number of appropriate applications for SAC technology which are not occurring because the folks with the application don't know what SAC can do and the folks with SAC don't know of the potential applications. This will be remedied indirectly and slowly by measures which increase awareness of SAC at the college level. For quicker results:

Let NSF solicit grants oriented toward applications of SAC technology.

A national SAC Center, which was amply engineering and application oriented, would help develop and help coordinate the development of applications of SAC technology.

Computer algebra clinic programs[1] would foster applications as well as promote undergraduate awareness of SAC. Let NSF solicit grants to launch or enhance computer algebra clinics.

---

[1] Described in (K) and (AA) in the: Notes on computational and computer algebra for U.S. science and engineering, in the preliminary reading material.

Moss Sweedler BIOGRAPHY April 17, 1988

Ancient history: Born '42; BS '63 MIT; PhD '65 MIT in Math-Algebra;
Professor, Mathematics '74 Cornell University

Recent history:

Visiting:	1987	IBM, TJ Watson Research Center, (summer)
Visiting:	1986-85	IBM, TJ Watson Research Center
Visiting:	1983	Ludwig-Maximilians-Universitaet, Munich
Fellowship:	1983	Alexander von Humboldt Foundation
Visiting:	1981	University of Tsukuba, Japan
Fellowship:	1981	Japan Society for the Promotion of Science
Fellowship:	1980	John Simon Guggenheim Foundation

FORMATIVE YEARS:

In elementary school, Chemistry was appealing, when the result was loud, bright or smelly. As an undergraduate, Math, Physics and EE (for both circuit design and computers) were appealing. Math won, after an inspiring undergraduate algebra course by Iwasawa.

PURITY (the pure math years):

My PhD and early research concern Hopf Algebras (HAs). HAs unify and generalize aspects of groups and Lie algebras. HAs led me to fundamental properties of purely inseparable field extensions. Properties whose proof did not require HAs and could have been discovered years before. H.Allen and I used these properties and further applications of HAs to settle a 25 year old conjecture of N.Jacobson.

Applications of HAs led me to work in: field theory, generalized Galois and descent theory, algebraic groups and geometry, commutative algebra, homological algebra, non-commutative algebra (especially representation theory), combinatorics (especially umbral calculus), differential algebra and other areas. Much of my work in these areas is independent of HAs.

BODY AND FLAVOR:

Within the algebra community I am best known for my body of work on HAs. I've written a book or so, and about 50 papers. From the beginning there has been a computational flavor to my work.

FORMATIVE YEARS II:

In the early seventies R.Larson and I used SNOBOL to generate examples in homological algebra. A pattern of zeros in the output suggested theorems we then proved. During '85-86, visiting the IBM Scratchpad group, I beheld a world of computer and computational algebra. A world known to remarkably few academic algebraists. While at IBM, experiments utilizing Scratchpad II, gave the key to the proof of the positive characteristic analog to the calculus classic: CLOSED IF AND ONLY IF EXACT. It plays a vital role in positive characteristic symbolic integration.

THE AFTERpureMATH:

I see areas of algebra emerging as applied mathematics, especially via computers, and see myself as an applied mathematician moving toward computer science. I've been active in the area of Groebner bases and may begin working on effective methods for real algebraic geometry. Another direction for future research is to use positive characteristic symbolic integration to help with characteristic zero symbolic integration, as is the case with factorization, a la Berlekamp.

Bruno Buchberger.

1986 - founder and director of RISC-LINZ
(Research Institute for Symbolic Computation Johannes Kepler Univ Linz)
1985 - editor-in-chief of Journal of Symbolic Computation (Academic Press)
1985 Chairman of the EUROCAL 85 conf (430 participants)
1984 Organizer of the ICME Session on "SC and math education"
1982 - design and implementation of special graduate curriculum in symbolic computation at J Kepler Univ
1974 - full professor of math (for comp scie) at J Kepler Univ
1973 habilitation in math (U of Innsbruck)
1966 - 1974 ass professor at U of Innsbruck
1966 Ph.D. in math (U of Innsbruck) (thesis in computer algebra: introduction of Grobner bases)

Books on Computer Algebra (ed. jointly with G.Collins and R.Loos), Math for Comp Scie (co-authored by F. Lichtenberger), Algorithms in Real Algebraic Geo (ed. jointly with D. Arnon).

Research Interests: algorithms in algebraic geometry, geo theorem proving, parallel symbolic computation, applications in robotics and CAD/CAM.

Bruno Buchberger.

Nr.1
Problems:

Symbolic computation problems arise only at relatively "high level" in applications. This implies that they arise normally only in quite large industrial companies, labs, institutions etc. Geographically, the groups in which the problems arise often are quite distant from the experts working in symbolic computation. Geographical distance often results in absence of communication between application people and symbolic computation experts.

Also, symbolic computation problems tend to occur sporadically. Most users, labs etc. do not think about installing and maintaining an expensive SC system "in stock" for the few occasions when they would need it. In the moment they really need it, however, they then do not have access. So they resort to manual methods or bypass the SC problem.

Symbolic computation systems and other software systems (e.g. numerics libraries) sometimes are hard to interface. Also, manual data input is sometimes an obstacle that prevents possible users from resorting to a symbolic computation software system.

Computational time and space complexity is another obstacle for using the more sophisticated methods of symbolic computation for real-world problems. The user just gets frustrated when he sees the seemingly most trivial examples run for hours.

The availability of symbolic computation methods and systems is not well-known among possible users (engineers, scientists etc.). The systems are hardly mentioned in the standard math, computer scie or natural and technical sciences curricula let alone integrated and actively used in the standard education.

Solutions:

Research centers in symbolic computations should be created with office space available for visitors from industrial companies etc. in order to host them for a couple of weeks for joint work on real problems. NSF could take a promoting role in the realization of such centers.

Symbolic computation systems should be systematically integrated with other software systems, in particular, for numerical computation, geometrical and graphical computation, text processing. This can be pursued in two ways: setting up libraries comprising numerical, symbolic, graphical etc. computation or designing integrated software systems for "scientific computation". NSF could, in particular, play an important role in organizing the creation of a comprehensive library of algorithms.

Actions should be taken in order to integrate SC into the math, CS and technical scie curricula, see the proposal of working group 3.1.4 at ICME 5 (1984).

Bruno Buchberger.

Nr.2

Problems:

Symbolic computation systems and other software systems (e.g. numerics libraries) sometimes are hard to interface. Also, manual data input is sometimes an obstacle that prevents possible users from resorting to a symbolic computation software system.

There are "too many" SC software systems around most of which evolved from software experiments and were expanded over the years. Only a few are designed as industrial products. Thus, although there are so many systems around, the range of topics covered remains quite stable over the last few years. The reason for this might be that new systems mostly are based on the incentive to create a more convenient user interface or to realize a more systematic programming style. The relatively small groups working on the realization of new systems then have a hard time to implement the new system features {\bf and} to incorporate the algorithm spectrum of existing systems. No energy is left for significantly expanding the algorithm spectrum. \item SC software systems, graphical software, textprocessing, numerical computation, logic computation, computer-aided software development tools should be made available in integrated systems for "scientific computation". A common algorithm base would be necessary for these areas.

On the other hand, it should be possible to extract smaller systems tailored to particular application areas from large integrated systems for "scientific computation".

Solution:

Governmental institutions like NSF could concentrate on establishing, maintaining and expanding an algorithm base for symbolic computation (in the wide sense of computer algebra, theorem proving, geometrical computation etc.).

Based on the (system independent) algorithm library, Software companies could concentrate on developing modern integrated software systems for symbolic computation with nice user interface, convenient programming environment, graphics interace, textprocessing interface, features for tailoring specialized subsystems for special applications areas.

Nr.3

Problems:

Good mathematical theory in the field of symbolic computation is based on good expertise in algorithm implementation and vice versa. It is intellectually hard and extremely time consuming two integrate both the mathematical and the software technological expertise in the same persons.

Symbolic computation is the field of algorithmic problem solution in areas where the objects involved have semantics (this may be seen as a "definition" of the term "symbolic objects"). As a tendency, problems involving objects having semantics are algorithmically unsolvable or inherently complex. This is the reason why, on the one hand, one feels that being able to solve one of these problems is marvelous and, on the other hand, one is frustrated by the bad performance. Also, practical acceptance of

complex algorithms is low by nature.

There is a lack of self-contained treatments of symbolic computation algorithms in text books etc.

Since symbolic computation is not integrated in undergraduate standard math and CS curricula, graduate students who want to specialize in symbolic computation need a long preparatory phase before they can produce original results.

Solutions:

It is urgent to set up special curricula for symbolic computation that aim at educating students who are literate in mathematical thinking as well as in implementation issues.

More utopically, but more fundamentally, I imagine that the standard curriculum of math will have to change drastically in the direction of smoothly integrating the algorithmic aspect as an indispensable component. In a certain sense all of math is symbolic computation.

Nr.4
Problems:

Both symbolic and numerical computation need a lot of time to get expertise. It is hard to organize a scientific life-time in such a way that both fields could be integrated in one person.

Also, the ways of thinking and, correspondingly, the type of people involved in symbolic and numerical computation seem to be fairly different.

At present, numerical libraries and symbolic software systems are very little connected. It is hard to set up a research environment for research in the integration of symbolic and numerical computation.

Solution:

It is again a question of establishing new curricula or changing existing curricula of math and CS.

Nr.5
Problems:

Symbolic computation in the narrow sense of "computer algebra" is too small and too special for provoking a broad interest. Most research groups in symbolic computation are too small to have a lot of impact.

The professional organizations are too much in the spirit of "mathematical societies", i.e. with little feeling for marketing, sales promotion, organization, standards etc. This is in sharp contrast to fields involving less math, for example, graphics. Typical mathematicians think that symbolic computation is low level math because, instead of proving, one is experimenting on the computer. We know that the opposite is true; algorithmic problem solution needs more math, deeper theorems, more involved proofs than inconstructive ("pure") math. On the other hand, typical computer

scientist think that symbolic computation is inpractical because we use deep math. Thus, symbolic computation obtains little appreciation from both sides and, at the same time, crucially depends on both fields.

This implies also that symbolic computation has constant problems in raising funds from sources governed by, either, typical mathematicians or typical computer scientists.

Solutions:

Symbolic computation must be seen in a broader context than just computer algebra, see the editorial of the J of Symbolic Computation. Broadening the scope is necessary both for theoretical and for software technological reasons.

SIGSAM, SAME etc. need more professional administration, promotion. ISSAC must strengthen its role as the central annual international symbolic computation conference. Also, a close interaction with the annual conferences in comp geo and comp logic should be achieved.

Integrated software systems for scientific computation (symbolic, graphical, numerical, logic computation) must be designed and implemented. A professional market analysis would be a prerequisite.

Nr.6

I think it is not advisable to separate the three aspects of symbolic computation

algorithms and theory
software systems
applications

because problems arising in one of the aspects often have their origin in a problem belonging to the other aspect. For example, little acceptance of SC software systems for applications in a particular area may have an origin in the high complexity of available algorithms in this area.

In addition, I think that the restriction of symbolic computation to computer algebra is one of the reasons why symbolic computation is less used than desirable. At the beginning of symbolic computation in the late fifties and early sixties an integration of computer algebra with, at least, automated theorem proving was intended. Meanwhile "symbolic computation" has more and more restricted its scope to computer algebra. For the future, the integration of all fields in which symbolic objects (objects with semantics) are treated will be essential both for theoretical and for practical purposes, see the editorial of the Journal of Symbolic Computation (Vol. 1/1, pp. 1-6).

Anthony C. Hearn

PROFILE

Dr. Hearn is a member of the Corporate Research Staff at the RAND Corpo-ration in Santa Monica, California. He received his early education in Australia before going to England to obtain a Ph. D. degree in Theoretical Physics from Cambridge University in 1962. From 1962 until 1964 he was a Research Associate in Physics at Stanford University, and returned there as an Assistant Professor in 1965 after a year at the Rutherford Labora-tory in England. In 1969 he joined the University of Utah as an Associate Professor in Physics, and was made Professor in 1971. From 1973 until 1980 he was Professor and Chairman of the Department of Computer Science. He joined RAND in July, 1980 as Head of the Information Sciences Depart-ment, a position he held until August 1984 when he assumed his present position. The RAND Corporation is a private, nonprofit institution engaged in research and analysis of matters affecting national security and the public welfare.

Dr. Hearn is best known for his pioneering work on the development of the algebraic manipulation program REDUCE which is used in over a thousand scientific institutions throughout the world. His current interests include algebraic manipulation, computer networking and scientific compu-tation. He has published many articles in the symbolic computation area including a manual for REDUCE and several reviews of the field. He is a member of the ACM, SIAM and the American Physical Society. He was a SIAM National Lecturer during the 1976-1977 academic year and an ACM National Lecturer from 1978 until 1980. He was an Alfred P. Sloan Fellow from 1967 until 1969.

The Future of Computer Algebra

A Draft Position Paper

Anthony C. Hearn
The RAND Corporation

21 April 1988

[For use during the workshop on Symbolic and Algebraic Computation: Directions for Future Research --- Not for further distribution]

On a number of occasions during the past year or so, I have given a talk entitled "The Future of Computer Algebra". I have been meaning to write this up for publication, but there are always a dozen other things that get in the way. However, I decided that the need for position statements for the forthcoming workshop was a good enough reason for me to summarize the main points I make in this talk (maybe this will also motivate me to write it up properly). I know that individual statements on various issues were asked for, but I hope noone will take umbrage at my alternative approach.

Point 1: By the mid-1990's, every serious scientist should have access to a personal workstation with a 20-50 MIPS processor, 16-32 megabytes of memory, at least a gigabyte of random access secondary storage, and access to high resolution graphics and printing capabilities. Any future algebra system design must take into account these hardware trends. It's a waste of time to design future systems that fit in 512 Kbytes of memory for example, or don't make proper use of bit mapped displays.

Point 2: Stand-alone production systems (such as Macsyma, Maple or Reduce in their present forms) will disappear. We shall see instead scientific and engineering environments that incorporate algebraic capabilities. However, the numerical, graphical, and text processing capabilities of these new environments will completely dominate the design. How should computer algebra system research be conducted in such a climate?

Point 3: In spite of Point 2, people will continue to design and build new algebra systems. The question is whether these people are doing science or not (since many people who do this work are in universities or research organizations). I maintain there are now only two scientific reasons for building new systems:

a) to support research in another field. This would probably be a specialized system, and any publication would emphasize the unique science done in the application field that was made possible with this system, not the system per se. This is much the same as an experimental physicist who designs a piece of apparatus to do an experiment. It's the experiment itself that's significant, not the apparatus.

b) to test new system design principles. The emphasis here is on *new*.

What is *not* research these days is the implementation of an existing algebra design (or a similar design) on a different computer or in a different language. That's technology or hacking, depending on your point of view. It's fine to do that if your goal is to have fun or make money, but it's *not* fine to do it if you want to get a degree or tenure from the activity.

Point 4: Algebraic algorithm development will have little influence on future algebra system design. I am *not* saying that such algorithm development is not important --- it is *very* important. However, because of the trends in hardware and software development, especially those I've indicated above, new algorithms will have to be adapted to system designs rather than the other way around. I therefore believe that we should think seriously about how to make that adaptation as effective as possible.

I may be wrong on this point. If custom chip production becomes as easy as making photographic copies, there may be a justification in designing a special purpose chip to do say Hensel lifting, or even something as broad as polynomial factorization. However, I just don't see the economics on our side for the foreseeable future.

Point 5: "Computer Algebra" is different from "Algebraic Computation". The analogy I use here is with "computer science" and "scientific computation". "Computer algebra" is what the algorithm and system developers do. It's the stuff that's reported at SIGSAM meetings and the like. "Algebraic computation" is what the *users* of algebra systems do. These people don't go to SIGSAM meetings. Instead, they go to their discipline-oriented meetings. In many cases, the use of an algebra system is not even mentioned in reporting the research (as one such person put it to me: you don't mention Fortran every time you produce a table of numbers). In other cases, an algebra system is used to get a result. The researcher often then uses the insight gained from the calculation to do it by hand, feeling in the process that this is a more intellectually dignified way to do it.

Other researchers use algebra systems to produce numerical code. I've estimated that over 50% of the weighted cycles of computing used to run REDUCE is spent producing Fortran code. However, how many papers at the typical SIGSAM meeting talk about doing a better job of this? Surely, given its importance, there should be more interest in the problem.

The question then is: do the "computer algebra" people produce what the "algebraic computation" people need? In fact, do we really know what they need? On the other hand, is this question relevant to the "computer algebra" community? Maybe we should just continue "doing our thing" and let the rest of the world catch up with us eventually.

Point 6: Education. Just as every researcher by the mid 1990's will have the sort of machine I mentioned at the beginning, every high school student will have a personal computer with at least 1 megabyte of memory (maybe 4), 100 megabytes of secondary storage, and a bit-mapped display. This could be a *huge* market for algebraic techniques. What shall we offer this community? The current systems with a different user interface? Some educators say they want to see the steps in the

calculation. Is this reasonable? In other words, should the algebra system *teach* how to do algebra, or is it just a calculator? My position on this is that I see algebraic techniques as being important, but again as part of a larger environment that may in fact include CAI.

Name: **John Fitch**

BIOGRAPHY

Student at St John's College, Cambridge (England) from 1964-1971. Mathematical Tripos 1964-67, BA 1967
Diploma in Computer Science with Distinction 1968, (MA 1970)
PhD 1971 for dissertion "An Algebraic Manipulator"
Senior Assistant in Research in the Cambridge Computer Laboratory 1971-1977 Adams' Prize co-winner 1975
Visiting Assistant Professor University of Utah 1977-78
Lecturer University of Leeds 1978-1980
Professor of Software Engineering University of Bath 1980-
Directory of Computer Service University of Bath 1980-1985

Partner in Fitch-Norman (LISP and Computer Algebra) 1983-
Director-designate, Codemist Ltd 1988-

Main research areas have been Computer Algebra, and LISP.
Author of CAMAL, contributor to REDUCE, applications in general relativity Co-author of Cambridge LISP.

Name: **John Fitch**

Position 1:

Not certain; partly information and wide availability on smallish machines. This is already happening. Other problems are handled later. User level programming still needs a certain way of thinking.

Position 2:

There are problems in the engineering in many cases. It takes a long time to translate algorithmic concepts into product-quality code. At present this relies on ad hoc methods. Looking longer term re-casting in say object form should lead to better systems...this is all very vague!

Position 3:

Many nice algorithms take too much space and time to run. There are holes in the understanding of say integration. I am not the correct guy to answer this.

Position 4:

The area has been neglected. There are good system for generating FORTRAN, but the need is for strong integration between the numerical libraries and the symbolic pre and post procesing. Part of the problems are interfaces. There was a short-lived product linking MACSYMA to the NAG libraries. This needs to b emuch more part of the main thinking in design. The switch between the two is a major canyon.

Position 5:

The provision of libraries and example codes for anything other than trivial examples is a great lack. There is an opportunity for integrating SAC within the teaching of math and engineering as a main tool. This means a re think of teaching throughout the system. There are signs of it happening.

Information systems on hints/examples?

Position 6:

The user interfaces are still old fashioned, and this relates to the issue of numerical-symbolic. The algebraic phase of an application is usually a small but hard part. This needs to be recognised.

NSF Should Solicit Proposals for Research in Computer Algebra

Position Paper

A Workshop on Symbolic and Algebraic Computation:
Directions for Future Research
April 28-29, 1988

Richard D. Jenks
IBM Thomas J. Watson Research Center
Yorktown Heights, New York 10598

This paper discusses how NSF can provide leadership towards bringing computer algebra into mainstream use in U.S. industry, university, and government environments. The number of people developing or using computer algebra in the U.S. is very small. We need to broaden this base dramatically. To do this, I recommend NSF solicitation of proposals for research in key areas of computer algebra. While such solicitation is not a customary policy of NSF, it is crucially important to achieve a needed influx of people into the field.

This paper is organized into four sections. Each describes one of four major problems with computer algebra systems today: accessibility, ease-of-use, breadth-of-focus, and education.

Accessibility.

Computer algebra systems on mainframes are notoriously unpopular with computer center managers and non-users because they are so space- and time-consumptive. While mainframes will continue to be used for large scale computations, personal machines will be preferable for most computer algebra. They are friendlier. There is no personal competition for cycles. Technology --and the commercial marketplace-- should eradicate the problem of accessibility within 3 years. The other problems however require NSF leadership and support to be solved.

Ease-of-Use.

Computer algebra systems today are not user friendly but can be made so by putting MathScribe- or Milo-style front ends on them and adding good graphics. A good user interface is crucial: the system should entice newcomers rather than turn them away.

I now propose two types of future computer algebra systems: "general purpose systems" and "special purpose systems." No examples exist of either kind today, as described, although some existing systems are perhaps a few man years away. A general purpose system is an engineering tool that can be casually used by non-specialists.
It may be viewed as a foolproof, "knowledge-based" system which provides a turn-key user interface for solving problems analogous to a high-level desk calculator. The system provides a broad range of sophistical computational machinery often required to solve even the most simply stated problem. And it does this by hiding all of the complexity from the non-specialist.

The general purpose system is also a tool for the specialist. It provides a general purpose programming language allowing users direct access to all system facilities. It ideally provides a computational model enabling a specialist to easily know how to solve any

problem the system is capable of solving. In short, the general purpose system contains a master library of every conceivable algebraic package and tool for computation. Since a good general purpose system is likely to become a commercial success, NSF support for its development is probably not appropriate.

The general purpose system attempts to do everything for all people. This displeases the specialist who just wants something for some people. But research is always carried out by specialists! A special purpose system designed exclusively for use in specialist areas is preferable. Examples include computational fluid dynamics, celestial mechanics, superconductivity, and various branches of mathematics such as computational topology. Each special purpose system has its own language allowing the specialist to use customary notations for that special area. But, and I want to emphasize this point, the special purpose system is produced as a extension and/or customization of the general purpose system.

An alternate approach to the special purpose system is to provide customized interfaces to computer algebra subsystems to make them accessible from other systems and languages such as FORTRAN.

There are few examples of computer languages in important specialist areas. A cooperative effort of the computer science community and specialists is needed to ease this problem. NSF should solicit proposals in this and related areas.

Breadth-of-focus.

The effort required to build a general purpose system is very large. As a result most computer algebra system development has centered on algebra not graphics and not numeric computation.

I digress briefly into a discussion of applications. Computer algebra can be helpful in any numerical area where models are describable by formulae. Symbolic computation can be used to transform expressions into forms which directly relate to the answer to the questions scientists are asking. Such problem transformations could give rise to much smaller numerical computations and considerable savings of CPU cycles. Unfortunately, today's computer algebra systems cannot do the numerical computational efficiently.

The symbolic and numeric communities are now very much separate. The attempt at integrating the NAG library of numeric facilities into MACSYMA has been disappointing.Numerical analysts generally want access to the code. Switching from MACSYMA to FORTRAN and back is most unpleasant. Morven Gentleman writes: "Inability, or even awkwardness, in fitting (numeric and symbolic languages together) is probably a large part of why symbolic algebra has been less useful to numerical analysts than might have been hoped."

One solution is to provide a coherent computational environment where both kinds of computation can be done with equal ease. Numerical primitives should be added to symbolic systems by competent numerical analysts. Numerical libraries could then be produced and reside in the master library of the general purpose system.

Computer graphics is also a tool of major importance for use in a general purpose system. Ease of movement between formulae and pictures is as important as between formulae and numbers (or between numbers and pictures).

NSF should solicit proposals for using or extending symbolic systems both for numerical

and graphical computation.

Education.

Computer algebra is now available on an HP calculator which was sold at the January AMS meeting for $60. To make general purpose computer algebra systems widely accessible on personal machines in the average university would require the total cost of hardware and software to be $1-$2K. We are about a factor of 5-10 or about 3 years away from that today. In time computer algebra will become a standard tool used by students everywhere. Yet we are not currently graduating students or producing teachers who have knowledge or skills in this area.

Computer algebra systems are now affordable for limited classroom use by all top universities. NSF should immediately solicit proposals for pilot programs involving computer algebra in education at the university level. These should come from educators not from the system developers who have more than enough to do.

Bio for B. F. Caviness

Bob Caviness is currently the NSF Program Director for Numeric and Symbolic Computation in the Division of Computer and Computation Research. He is on a one-year leave of absence from the University of Delaware, where he is Professor of Computer and Information Sciences with a joint appointment in Mathematical Sciences. He is the immediate past Chair of the Department of Computer and Information Sciences.

His research interests are in symbolic and algebraic computation, particularly decision procedures for determining closed form solutions of differential equations and more recently in designing algebraic algorithms for new computer architectures. Currently, with Robert Gilbert and Roman Shtokhamer, he is writing a textbook titled "Applied Symbolic Computation Using Macsyma."

He is an associate editor of the Journal of Symbolic Computation, the area editor for symbolic manipulation for the ACM Computing Reviews, and a past Chair of the ACM Special Interest Group on Symbolic and Algebraic Manipulation (SIGSAM).

His PhD degree is in computer science from Carnegie-Mellon University.

A Workshop on Symbolic and Algebraic Computation:
Directions for Future Research

Position Statement

Name: Bob Caviness

Topic: A debate on why NSF gets so few proposals in symbolic and algebraic computation. Or is this area really important?

Proposition for debate: Fewer than 15% of the proposals received by the NSF Program for Numeric and Symbolic Computation prior to FY88 (in FY88 27% or 20 of the 74 new proposals received to date are in symbolic and algebraic computation) have been on symbolic and algebraic computation. In general, few proposals are received Foundation-wide in this area. If this research area has so much potential and intellectual interest, why isn't this expressed by ferment in the scientific community and subsequently more proposals to the Foundation? Because NSF has not funded good proposals, ... NSF ... Why? NSF reviewers have not done a good job. Why? Bad proposals

Response: Understanding the reasons why there has not been a more vigorous research community is this area is an important key in deciding how to accelerate research. Some contributing factors are:

First, research in symbolic and algebraic computation requires a significant integration of sophisticated knowledge and abilities from both computer science and mathematics. On the mathematics side algebra is the key area, but analysis, especially that analysis related to applied mathematics, complex variables, algebraic geometry, group theory, number theory and logic are often important. In computer science, important areas are the theory of algorithms, complexity theory, programming language design and implementation, large-scale system design and implementation, and heuristic programming. In addition, it is useful to know traditional numerical computation. This is a broad spectrum of knowledge even without applications areas, but not an impossible one. Further there are other intellectually vibrant research areas that require equally broad or broader knowledge. So this alone cannot explain the low level of activity.

But more important than the breadth of the spectrum, is the fact that it encompasses a range of topics that are rarely tied together in our (here "our" means US; some other countries do a much better job here) educational programs. Furthermore as computer science has matured as a discipline, a healthy process, one of the unfortunate aspects has been that it has grown away from mathematics. Twenty years ago, graduate students studying computer science learned much more mathematics than they do today. Many of the current persons working in symbolic and algebraic computation were educated when computer science and mathematics were more closely linked or else were educated in Europe or other places where those links remain in place (at least more so than in the US). Hence, in the last few years few students have been well-educated for research in this area, and this is one reason for the relatively low level of activity in the US.

It was not until the mid-1960's that there were computers capable of doing substantial algebraic computation. So one sees little activity in this area prior to 1965. This is in contrast to research in numerical computing that was well underway by the end of WWII, at least twenty years earlier. In further comparison with numerical computing, the last twenty years have seen a significant improvement in the hardware and software platforms for research in numerical computing. An important improvement on the software side has been the establishment of high-quality, robust libraries; on the hardware side,

supercomputers have been designed for numerical computing. The total effect has been to make numerical computing a much more powerful tool and to help contribute to a more vigorous research community. The analogous platforms are only partially in place on the symbolic side, although workstations designed for symbolic computing, large-scale architectures like the connection machine, and languages like Common Lisp may be setting the stage for real progress that can build in more effective ways on the existing software for algebraic computation.

So two key reasons for the low level of activity is the lack of programs to educate students for research in this area and the lack of a sufficient research platform in software and hardware (the former is a more substantial problem than the latter) to facilitate development of systems for symbolic computation.

To solve the first problem both short- and long-term action is needed. For the short term, possibilites include programs to attract post-PhDs into this area via special years, workshops, etc. For the long-term interdisciplinary graduate programs between computer science and mathematics need to be nurtured.

To improve the research platform in this area, a plan is needed that outlines what is needed in the way of new software and hardware. Possible software components are libraries of algorithms, user interface components, and other tools that make it possible to build special-purpose systems with a minimum of effort. Given a blueprint of what's needed, an action plan will be needed to facilitate the building of the platform.

Shreeram Abhyankar
Purdue University
Department of Mathematics
Purdue University
West Lafayette, IN

47907
Tel. 317-743-2440
e-mail ram@cs.purdue.edu

William Adams
National Science Foundation
1800 G. Street,N.W.
Washington, D.C.

20550
Tel.
e-mail

Richard Askey
University of Wisconsin
Department of Mathematics
480 Lincoln
Madison, Wisconsin

53706
Tel. 608-262-1234
e-mail

Peter Bates
National Science Foundation
1800 G. Street, N.W.
Washington, D.C.

20550
Tel. 202-357-3686
e-mail pbates@note.nsf.gov

Ann Boyle
National Science Foundation
1800 G. Street, N.W.
Washington, D.C.

20550
Tel.
e-mail aboyle@note.nsf.gov

Charles Brownstein
National Science Foundation
National Science Foundation
Computer & Information Science & Engineerin Rm 306
1800 G Street, N.W.
Washington, D.C.
20550
Tel.

e-mail

Bruno Buchberger
RISC-LINZ
Johannes Kepler Universitaet
A-4040 Linz
AUSTRIA

Tel. 43-732-2468 ext: 9219
e-mail K313370@aearn.bitnet

B. F. Caviness
National Science Foundation
1800 G Street, N.W.
Washington, D.C.

20550
Tel.
e-mail bcavines@note.nsf.gov

Jagdish Chandra
U. S. Army Research Office
P. O. Box 12211
Research Triangle Park, NY

27709-2211
Tel.
e-mail

Ray Chin
National Science Foundation
1800 G. Street, N.W.
Washington, D.C.

20550
Tel.
e-mail rchin@note.nsf.gov

David Chudnovsky
Columbia University
Math Department
Columbia University
New York, NY

10027
Tel. 212-864-5320
e-mail

Melvin Ciment
National Science Foundation
National Science Foundation
Division of Advanced Scientific Computing, Rm 417
1800 G Street, N.W.
Washington, D.C.
20550

e-mail

Andre Deprit
National Bureau of Standards
19119 Norman Way
Gaithersburg, MD

20879
Tel. 301-963-2161(b) 975-2709 (o)
e-mail

David Elliott
National Science Foundation
Systems Theory & Operations Research Program
Room 1151
1800 G Street, N.W.
Washington, D.C.
20550
Tel. 202-357-9618
e-mail delliott@nsf or delliot@note.nsf.gov

Richard Fateman
Univ. Calif. Berkeley
EECS-CS Division
573 Evans Hall
Berkeley, CA

94720
Tel. 415-642-1879
e-mail fateman@renoir.berkeley.edu

John P. Fitch
University of Bath
School of Mathematics
University of Bath, Claverton Down
Bath, BA2 7AY
Great Britain

Tel. 011-44-225-826820 (o) 011-44-225-837430 (h)
e-mail jpff@maths.bath.ac.uk

Peter Freeman
National Science Foundation
Rm 304
1800 G Street, N.W.
Washington, D.C.

20550
Tel. 202-357-9747
e-mail pfreeman@note.nsf.gov

Marvin (Marven) Gentleman
National Research Council of Canada
Division of Electrical Engineering
Montreal Rd. Bldg M-50
Ottawa (K1A0R8)
Ontario, Canada

Tel. 613-998-3857

Gaston H. Gonnet
University of Waterloo
Department of Computer Sciences
Waterloo, Ont. NZL 3G1
Canada

Tel. 519-888-4460
e-mail ghgonnet@waterloo (csnet)

Gerald Guralnick
Brown University
Department of Physics
Box 1843
Providence, RI

020912
Tel. 401-863-2623
e-mail gerry@csc.brown.edu

Anthony Hearn
Rand Corporation
1700 Main Street
Santa Monica, CA

90406-2138
Tel. 213-393-0411 X6615
e-mail

Harry Hedges
National Science Foundation
Rm 304
1800 G Street, N.W.
Washington, D.C.
20550
Tel.
e-mail hhedges@note.nsf.gov

Richard Hirsh
National Science Foundation
1800 G Street, N.W.
Washington, D.D.

20550
Tel. 202-357-9776
e-mail rhirsh@note.nsf.gov

Kenneth Hoffman
Office for Governmental and Public Affairs
Joint Policy Board for Mathematics
Office for Governmental and Public Affairs
818 Connecticut Avenue
Washington, D.C.
20006
Tel.
e-mail

Richard Isaacson
National Science Foundation
Division of Physics
1800 G. Street, NW
Washington, D.C.

20550
Tel.
e-mail

Richard Jenks
IBM
T.J. Watson Research Center
P.O. Box 218
Yorktown Heights, NY

10598
Tel. 914-945-1233
e-mail jenks@ibm.com (csnet) or jenks@yktvmz (bitnet)

William Lakin
National Science Foundation
Division of Mathematical Sciences, Rm 339
1800 G Street, N.W.
Washington, D.C.

20550
Tel. 202-357-3686
e-mail wlakin@note.nsf.gov

George Lea
National Science Foundation
1800 G Street, N.W.
Washington, D.C.

20550
Tel. 202-357-7737
e-mail glea@note.nsf.gov

Michael Levine
Pittsburg Super Computer Center
Carnegie-Mellon Institute
ATTN: Elvira Prologo
Pittsburg, PA

15213
Tel. 412-268-2671/4960
e-mail levine%cpwscb@morgul.psc.edu

Andrzej Manitius
National Science Foundation
National Science Foundation, Rm 339
1800 G Street, N.W.
Washington, D.C.

20550
Tel.
e-mail

James McIver
SUNY Buffalo
Department of Chemistry
Buffalo, NY

14214
Tel. 716-831-3629
e-mail mciver@sunybcs.bitnet

Anil Nerode
Mathematical Sciences Institute, Cornell Univ.
Mathematics Sciences Institute
Cornell University
201 Caldwell
Ithaca, NY
14853
Tel.
e-mail

Alan Perlis
Yale University
Department of Computer Science
P.O. Box 2158 Yale Station
New Haven, CT

06520
Tel. 203-436-8160
e-mail

Louis Raphael
National Science Foundation
1800 G Street, N.W.
Washington, D.C.

20550
Tel.
e-mail

John Rice
Purdue University
Department of Computer Sciences
West Lafayette, IN

47907
Tel. 317-494-6003
e-mail jrr@cs.purdue.edu

William F. Schelter
University of Texas
Department of Mathematics
University of Texas
Austin, TX

78712
Tel. 512-471-1141
e-mail wfs@carl.ma.utex.as.edu

Carl Smith
National Science Foundation
National Science Foundation, Rm 305
1800 G Street, N.W.
Washington, D. C.

20550
Tel.
e-mail

Philip W. Smith
IMSL
IMSL
2500 CityWest Blvd.
Houston, TX

77042
Tel. 713-782-6060
e-mail ...uunet!imsl!smith or imsl!smith@uunet.uu.net

Judith Sunley
National Science Foundation
National Science Foundation
Division of Mathematical Sciences
1800 G Street, N.W.
Washington, D.C.
20550
Tel. 202-357-9669
e-mail jsunley@note.nsf.gov

Moss Sweedler
Cornell University
Department of Mathematics
320 White Hall
Ithaca, NY

14853
Tel. 607-255-4373
e-mail jc5j@cornella

Alvin I. Thaler
National Science Foundation
National Science Foundation, Rm 304
1800 G Street, N.W.
Washington, D.C.

20550
Tel. 202-357-7349
e-mail thaler@broadway.expres.nsf.gov or thaler@nsf

Ralph Wachter
Office of Navel Research
Code 1133
Software Research Program
800 N. Quincy Street
Arlington, VA
22217-5000
Tel. 202-696-4304
e-mail rfw@aplvax.arpa or wachter@radmis-onr.arpa

Elbert Walker
National Science Foundation
1800 G Street, N.W.
Washington, D.C.

20550

Tel.
e-mail

Paul Wang
Kent State University
Kent State University
Kent, OH

44242
Tel. 216-672-2249
e-mail pwang%kentvax.csnet@relay.cs.net

Peter Weinberger
AT&T Bell Laboratories, Inc.
600 Mountain Ave.
Murrary Hill, NJ 07974

Tel. 201-582-7214
e-mail pjw@research.att.com

Zeke Zaclstein
National Science Foundation
Division of Computer &
Computation Research, Rm 304
1800 G Street, N.W.
Washington, D.C.
20550

Tel.
e-mail